Technická Univerzita v Liberci

Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2612 Elektrotechnika a informatika

Studijní obor: 1802R022 Informatika a logistika

Skript pro import 3D dat do programu Blender

Script for importing 3D data to Blender

Bakalářská práce

Autor: Zbyněk Hlava

Vedoucí práce: Ing. Jiří Hnídek

V Liberci 20.5.2010

Zadání

vložit originální zadání.....

Zadani

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mojí bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. O právu autorském, zejména § 60 - školní dílo.

Beru na vědomí, že Technická Univerzita v Liberci(TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřeby TUL. Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím projektu.

V Liberci dne _____ _____

podpis

Poděkování

Děkuji především vedoucímu této práce Ing. Jiřímu Hnídkovi za trpělivost a pomoc při řešení problémů. Dále Ing. Martinu Živnému za dodávání potřebných podkladů a Doc. Ing. Antonínu Potěšilovi Csc. za motivaci a pomoc při psaní této práce. Také bych chtěl poděkovat své rodině za možnost studia a za podporu.

Abstrakt

Cílem předložené práce bylo naprogramovat importní skripty pro vybrané 3D datové formáty do aplikace Blender. Prostředí Blender bylo vybráno na základě rešerše. Práce se věnuje části vývoje aplikace v rámci projektu „Inovace technologie výroby umělých kůží“ firmy LENAM s.r.o. Pro import byly vybrány formáty: NASTRAN, IGES a STEP, se kterými bylo třeba se nejprve seznámit. Skripty byly naprogramovány v jazyce Python. K importu IGES a STEP byla využita knihovna pythonOCC. Na závěr proběhlo měření rychlosti importu souborů typu NASTRAN.

Klíčová slova

Blender, import, NASTRAN, STEP, IGES

Abstract

Aim of this work was to create importing scripts for chosen 3D data formats to Blender, which was selected on result of a search. This works is a part of project 'Inovation of production technologies for creating imitation leather' for LENAM s.r.o. Following formats were chosen for import: NASTRAN, IGES and STEP. It was necessary to learn more about them first. Scripts were programmed in Python, which is scripting language in Blender. IGES and STEP files were imported with help of pythonOCC library. There was a NASTRAN importing speed test on the end.

Keywords

Blender, import, NASTRAN, STEP, IGES

Obsah

Seznam symbolů zkratek a termínů

CAD - Computer Aided Design

CAE - Computer Aided Engineering

OCC - Open CASCADE Technology

pythonOCC - pythonOpenCASCADE

GNU GPL - GNU General Public License

GNU LGPL - GNU Lesser General Public License

BSD - Berkeley Software Distribution

UI - User Interference

NASTRAN - NASA Structural Analysis

STEP - Standard for the Exchange of Product model data

IGES - Initial Graphics Exchange Specification

Kapitola 1

Úvod

Cílem této bakalářské práce bylo vytvořit importní skripty pro vybrané (3D)datové formáty. Skripty budou následně použity v softwarové aplikaci, která vychází z programu Blender a je vyvíjená týmem řešitelů z firmy LENAM, s.r.o., Technické univerzity v Liberci a firmy Magna Exteriors & Interiors (Bohemia), s.r.o. v rámci projektu MPO TIP 2009 „Inovace technologie výroby umělých kůží, ev.č. FR - TI1/266“.

Prostředí Blender bylo vybráno na základně rešerše provedené za účelem najít nejvhodnějšího kandidáta pro danou aplikaci. Určující byla kritéria, která jsou charakterizována v kapitole 2.2.

V práci jsou popsány možnosti využitých softwarových nástrojů a programových prostředí Blender, Python, Open CASCADE Technology a pythonOCC, které byly použity v rámci řešení zadání práce. Obsaženy jsou rovněž datové formáty vybrané k importu a jednotlivé importní skripty.

Kapitola 2

Projekt

2.1 Úvod

Cílem výše zmíněného MPO projektu je vytvořit aplikaci schopnou simulovat radiační ohřev forem v procesu vytváření umělých kůží. Na projektu spolupracuji s kolegou Bc. Janem Quaiserem, který se zabývá některými dalšími funkcemi plánované aplikace. Naše aktivity a podíl na řešení je řízen panem Ing. Martinem Živným, který dodává podklady k práci a dohlíží na postup provedených prací.

Prvním krokem byla rešerše výběru vhodného softwaru, která měla vytipovat aplikaci jež bude dále upravována do podoby jakou si tým řešitelů projektu představuje.

2.2 Rešerše a analýza

Jako možné kandidáty na realizaci softwaru byly zvažovány a hodnoceny následující svobodné softwary: Blender, FreeCAD, Heekscad, k-3D a Salome. Současně byla definována kritéria, podle kterých byly tyto aplikace hodnoceny.

2.2.1 Kritéria

1. kritérium - licence V našem případě nemělo toto kritérium přílišnou váhu, jelikož jsou všechny porovnávané aplikace volně dostupné. Přesto jsou v licencích drobné rozdíly, které mohou ovlivnit hodnocení.

Blender	GNU GPL
FreeCAD	GNU GPL & LGPL
heekscad	BSD
k-3D	GNU GPL
Salome	LGPL

2. kritérium - zaměření aplikace Toto kritérium hodnotilo, jestli se jedná o aplikaci zaměřenou na práci s CAD(Computer-aided design) daty, či naopak aplikace obsáhne širší pole funkcí. Pro náš účel by byla nejvhodnější kombinace obou případů, jelikož pracujeme s CAD modely a zároveň simulujeme ozáření. Bylo by tedy dobré aby aplikace zvládala výpočty ozáření a následné zobrazení průběhu zahřívání, ale také aby bylo možné upravovat modely případně nastavovat hodnoty pro zářiče, a tak dále.

Blender	komplexní aplikace pro tvorbu 3D grafiky
FreeCAD	CAD/engeneering
heekscad	CAD
k-3D	víceúčelová aplikace pro tvorbu 3D grafiky
Salome	CAD/CAE

3. kritérium - kvalita kódu Pro vývoj a úpravy aplikace je důležitá kvalita stávajícího kódu, zda v něm autoři komentují průběh, případně zapisují poznámky pro další vývoj daných částí programu. Jelikož jsme se snažili najít software, který by bylo možné v případě potřeby více či méně upravit, je dobré mít kvalitní zdrojový kód a komentáře od autora, které mohou práci značně ulehčit.

Blender	časté hodnotné komentáře
FreeCAD	časté komentáře
heekscad	bez komentářů
k-3D	komentáře jen zřídka
Salome	bez komentářů

4. kritérium - dokumentace Pro práci v té či oné aplikaci je třeba jí rozumět, to samé platí pro její vývoj. Bylo tedy důležité, aby měl software dobrou dokumentaci, která nám v případě problému může pomoci.

Blender	velmi rozsáhlá (také česky), spousta tutoriálů
FreeCAD	dobrá, obsahuje tutoriály pro psaní skriptů
heekscad	zatím neexistuje
k-3D	dobrá
Salome	dobrá

5. kritérium - komunikace s uživateli Pokud je zapotřebí odbornější pomoci, je dobré mít možnost kontaktovat zkušenější uživatele, či přímo vývojáře aplikace. Často se tím zkrátí čas vývoje. Proto se v rešerši objevilo i následující kritérium.

Blender	rozsáhlá diskusní fóra (také česká), IRC, weby věnované výhradně Blenderu
FreeCAD	oficiální fórum (cca. 600 příspěvků)
heekscad	Google group(kolem 25 členů), autorův blog
k-3D	oficiální fórum (cca. 600 příspěvků), dotazy vývojářům, blogy
Salome	oficiální fórum (cca. 3000 příspěvků)

6. kritérium - využitelné funkce Důležité bylo jaké, za naším účelem, využitelné funkce softwary již obsahují. V našem projektu jsou důležité především tyto funkce:

- ovládání zobrazení
- manipulace s objekty a jejich editace
- kontrola průniků
- import CAD formátů
- export VRML případně obrázků
- možnost výpočtu/zobrazení průběhu ozařování

Blender	základní, + render, import STL, export VRML, snadná editace síťového modelu, práce se světly
FreeCAD	základní, + otevření IGS, STEP, STL, částečně NAS
heekscad	základní, + import IGS, STEP, STL, měření vzdálenosti(úhlů),SelectSimilar
k-3D	základní, + render, editace síťového modelu, práce se světly
Salome	základní, + import IGS, STEP, měření vzdálenosti(úhlů), skalární mapy

7. kritérium - rozšiřitelnost Pokud program dané funkce neobsahuje, velice záleží na tom, jaké má uživatel možnosti jejich přidání či úpravy do požadované podoby.

Blender	pomocí Python skriptů lze změnit UI, rozšířit import, funkce, render,...
FreeCAD	pomocí Python skriptů, lze rozšiřovat i UI
heekscad	pomocí Python add-in modulů
k-3D	Python skripty, K-3D skripty, možná tvorba pluginů
Salome	pomocí Python a C++ modulů

8. kritérium - platforma Jedna z dalších věcí, které jsou pro vyvíjený software důležité je jeho „přenositelnost“ mezi platformami. Tedy na jakých operačních systémech může aplikace fungovat.

Blender	Win, Linux/Unix, Mac OSX, Solaris, BSD
FreeCAD	Win, Linux/Unix, Mac OSX
heekscad	Win, Linux/Unix
k-3D	Win, Linux/Unix, Mac OSX, BSD, Solaris
Salome	Linux/Unix

2.2.2 Ohodnocení

Jednotlivá kritéria měla různou váhu v závislosti na důležitosti pro vývoj. Jelikož bylo třeba aplikaci značně upravit, nejdůležitějším kritériem byla rozšiřitelnost a komunikace s uživateli, případně vývojovým týmem. Jejich pomoc může často zjednodušit problém, případně ho i vyřešit. Nejméně podstatná byla kritéria, která nemají na vývoj až takový vliv, jako je licence (všechno jsou svobodné softwary).

aplikace	K1	K2	K3	K4	K5	K6	K7	K8
Váha	0,05	0,10	0,09	0,15	0,25	0,15	0,30	0,09
Blender	0,75	0,90	0,95	1,00	1,00	0,70	1,00	1,00
FreeCAD	0,85	0,65	0,85	0,75	0,50	0,60	0,70	0,70
heekscad	1,00	0,60	0,45	0,00	0,40	0,65	0,60	0,50
k-3D	0,75	0,85	0,75	0,90	0,60	0,65	0,60	0,80
Salome	0,80	0,75	0,65	0,85	0,75	0,70	0,40	0,25

Tabulka 2.1: Váha a ohodnocení kritérií

aplikace	K1	K2	K3	K4	K5	K6	K7	K8
Blender	0,04	0,09	0,09	0,15	0,25	0,11	0,30	0,09
FreeCAD	0,04	0,07	0,08	0,11	0,13	0,09	0,21	0,06
heekscad	0,05	0,06	0,04	0,00	0,10	0,10	0,18	0,05
k-3D	0,04	0,09	0,07	0,14	0,15	0,10	0,18	0,07
Salome	0,04	0,08	0,06	0,13	0,19	0,11	0,12	0,02

Tabulka 2.2: Výsledné hodnoty

2.2.3 Výsledky

aplikace	výsledná hodnota	pořadí
Blender	1,02	1
FreeCAD	0,71	3
heekscad	0,53	5
k-3D	0,76	2
Salome	0,68	4

Tabulka 2.3: Výsledky a pořadí

V tabulce 2.3 jsou výsledky porovnání jednotlivých aplikací a jejich výsledné pořadí. Podobnou rešerši provedl i kolega Bc. Jan Quiser se stejným vítězem.

Zvolena byla nejnovější verze Blenderu a to Blender 2.5, jelikož má o proti předchozím vydáním plně definovatelný interface. Blender 2.5 využívá pro tvorbu a zpracování skriptů nejnovější verzi programovacího jazyka Python. Jedná se o Python 3.1.

Kapitola 3

Teorie

3.1 Blender

3.1.1 Popis

Blender je multiplatformní OpenSource aplikace zaměřená na vytváření 3D modelů, animací, rendering, postproduction a v neposlední řadě tvorbu na interaktivních aplikací.

Multiplatformní neznamená nic jiného, než že má Blender verze dostupné pro operační systémy Windows, Linux, Mac OS a další. OpenSource znamená, že je program nejen zcela zdarma a to i pro komerční využití, ale také že si můžete stáhnout kompletní zdrojové kódy, zkompilovat je pro optimalizaci výkonu, upravovat a případně se aktivně podílet na dalším vývoji Blenderu.

Blender je možné rozšiřovat pomocí Python skritů, možností rozšíření je mnoho, počínaje importem datových formátů, přes změnu interface až po generování objektů. Toho lze dobře využít a velice si usnadnit práci.

3.1.2 Vývoj

Blender byl vytvořen holandskými studii NeoGeo a Not A Number (NaN). Za strůjce Blenderu je považován Ton Roosendaal, který v roce 1998 založil firmu NaN, aby mohl Blender nadále vyvíjet a rozšiřovat. Až do roku 2002 byl Blender komerční aplikací, bohužel v onom roce firma NaN zkrachovala. Věřitelé se dohodli, že se Blender stane svobodným softwarem distribuovaným pod licencí GNU GPL. Blender je teď aktivně vyvíjen díky komunitě uživatelů a Blender Foundation.

3.1.3 GNU GPL

GNU General Public License je licence pro svobodný software, původně napsaná Richardem Stallmanem pro projekt GNU. GPL je nejpopulárnějším a dobře známým příkladem silně copyleftové[1] licence, která vyžaduje, aby byla odvozená díla dostupná pod toutéž licencí. V rámci této filozofie je řečeno, že poskytuje uživatelům počítačového programu práva svobodného softwaru a používá copyleft k zajištění, aby byly tyto svobody ochráněny, i když je dílo změněno nebo k něčemu přidáno.[2]

Z toho vyplývá, že je možné software volně upravovat, díky čemuž také mohla vzniknout tato práce. Znamená to také, že skripty naprogramované v rámcí této práce mohou, ale nemusí být volně dostupné pro ostatní uživatele.

3.2 Python

Jedná se o interpretovaný programovací jazyk, tj. vykonává instrukce zapsané v kódu přímo. Opakem je překladač, který program nejprve přeloží do strojového kódu a teprve pak je možné ho spustit. Python v roce 1990 navrhl Guido van Rossum a je vyvíjen jako OpenSource projekt, který nabízí instalační balíky pro většinu dostupných platforem. Jazyk byl pojmenován podle „Monty Python's Flying Circus“, jehož je autor fanouškem. Při vývoji byl kladen důraz na tyto cíle:

- snadný a intuitivní jazyk, avšak dostatečně silný
- otevřený kód
- srozumitelný jako běžná mluva(angličtina)
- vhodný pro běžné úkoly, umožňující vývoj v krátkém čase

K význačným vlastnostem jazyka Python patří jeho jednoduché osvojení. Někteří ho dokonce považují za jeden z nejvhodnějších programovacích jazyků pro začátečníky. Je to dáno tím, že jedním z jeho inspiračních zdrojů byl jazyk ABC, který byl jako jazyk pro výuku přímo vytvořen. Python ale zároveň bourá zažitou představu, že jazyk vhodný pro výuku není vhodný pro praxi. Podstatnou měrou k tomu přispívá čistota a jednoduchost

[1] slovní hříčka ke spojení copyright

[2] dostupné z WWW: cs.wikipedia.org/wiki/GNU_General_Public_License/

syntaxe, na kterou se při vývoji jazyka dbalo. Python je také využíván jako skriptovací jazyk v aplikacích jako je Blender, Maya, Photoshop, atd.

3.3 Open CASCADE Technology

Open CASCADE Technology(OCC) je velice rozsáhlý projekt, zaměřený především na práci s CAD daty. Toho bude využito při importu IGES a STEP formátu, jelikož Blender nepodporuje tento způsob reprezentace 3D dat.(Více o způsobech reprezentace v kapitole 3.5.1).

Převážná většina freewarových CAD aplikací využívá OCC jako základ pro svojí činnost, proto byla zvolena ku pomoci při řešení importu některých datových formátů.

3.4 pythonOCC

Jde o Python verzi Open CASCADE Technology. Jelikož Blender využívá pro skriptování Python, přímé použití OCC pro psaní skriptu není možné. PythonOCC je, v tomto případě velice ceněný, modul pro Python vytvořený z knihovny OCC. Zatím nezahrnuje všechny funkce původní verze, avšak ty které budou třeba ke zdárnému vyřešení importu součástí knihovny jsou. Hlavní problém je v tom, že pythonOCC je dostupný pouze pro starší verzi Pythonu 2.x.

Z toho plyne, že jej není možné využít v Blenderu 2.5, který byl na základě rešerší zvolen jako základ vyvíjené aplikace. Skripty pro import těchto dvou datových formátů byly tedy naprogramovány pro starší verzi Blenderu 2.49, ta využívá pro práci se skripty vhodnou verzi jazyka Python, což není v rozporu se zadáním práce.

3.5 Datové formáty

Pro import byly vybrány následující tři datové formáty:

- NASTRAN
- STEP

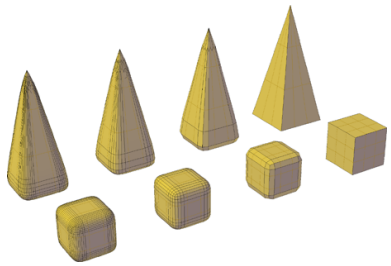
- IGES

Jedná se o základní formáty, se kterými firma LENAM s.r.o. pracuje a Blender jejich import zatím nepodporuje.

3.5.1 Mesh, Solid, Surface

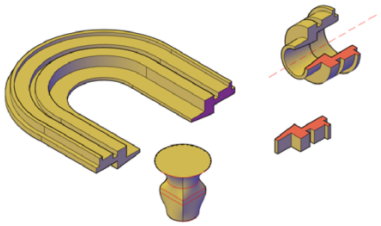
Pro snazší pochopení dalšího obsahu práce jsou zde popsány tři základní typy reprezentace 3D dat. Jedná se v podstatě o to z čeho je objekt vytvořen případně uložen v souboru.

Mesh - tvar geometrie je jednoznačně určen pomocí uzlů(vertices), hran(edges) a ploch(polygons/faces). Takto reprezentuje 3D data spousta grafických softwaru jako třeba právě Blender.



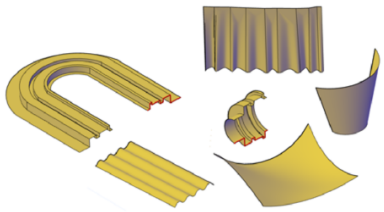
Obrázek 3.1: Ukázka geometrie mesh

Solid - tvar objektu je určen na základě boolean operací mezi jednotlivými geometrickými tvary. Objekt je „plný“ (solid) nejedná se tedy pouze o reprezentaci tvaru, ale hmoty, proto je možné na objektech testovat různé reálné situace.



Obrázek 3.2: Ukázka geometrie solid

Surface - tvar plochy je dán na základě několika určujících křivek. Jedná se například o tzv. NURBS(Non-uniform rational basic spline) modelování, které využívají aplikace jako Rhinoceros3D.



Obrázek 3.3: Ukázka geometrie surface

3.5.2 NASTRAN

NASTRAN (NASA Structural Analysis) je z trojice vybraných formátů nejjednodušší na zpracování, protože se jedná o formát typu mesh. To je pro import do Blenderu výhoda, jelikož jsou v něm 3D data reprezentována stejným způsobem.

Formát NASTRAN má dva hlavní způsoby zapisování dat, záleží na aplikaci, ve které byl soubor uložen, skript umí zpracovat obě.

Z níže uvedeného kódu je vidět že slovem *GRID* respektive *GRID** jsou označeny řádky na kterých je pozice jednotlivých uzlů v kartézské soustavě souřadnic(X,Y,Z). Druhá sekce určuje plochy případně samostatné hrany. *CQUAD4* je označení pro čtyřúhelník, *CTRIA3* pak označuje trojúhelníky a logicky *CBAR* uvozuje řádek s informací o samostatných hranách. Další části kódu jsou v případě této práce irelevantní, jelikož nám jde pouze o import geometrie objektu.

Zde je příklad formátu NASTRAN:

```
BEGIN BULK
$---5---10----5---20----5---30----5---40----5---50----5---60----5---70----5---80
$ NODAL COORDINATES ============================================================
$        NODE_ID               X       Y       Z
GRID           1              1.      0.      0.
GRID           2              1.      2.      0.
GRID           3              3.      2.      0.
GRID           4              3.      0.      0.
GRID           5              2.      4.      0.
GRID           6              2.      6.      0.
GRID           7              1.      6.      2.
GRID           8              0.      2.      0.
GRID           9              0.      3.      0.
GRID          10              0.      3.      1.
GRID          11              0.      2.      1.
$ ELEMENT CONNECTIVITY =========================================================
$         ELE_ID GEOM_ID  NODE_1  NODE_2  NODE_3  NODE_4
CQUAD4         1       1       1       2       3       4
CTRIA3         2       1       5       6       7
CBAR           3       3       2       5
CQUAD4         4       2       8       9      10      11
$ GEOMETRY =====================================================================
$ tj. definice partu(PID)/geom. vlastnosti/skupin
$HMNAME COMP                    1"PART_1"
$HMCOLOR COMP                   1      10
PSHELL         1              1.
$HMNAME COMP                    2"PART_2"
$HMCOLOR COMP                   2      13
PSHELL         2              2.
$HMNAME PROP                    3"PART_3"
PBAR           3       0      4.
$ MATERIALS ====================================================================
$ tady zadne nejsou
ENDDATA
```

Hlavním rozdílem druhé možnosti záznamu dat je uložení souřadnic uzlů, kde je souřadnice Z uložena na samostatném řádku, který je uvozen znakem $*$.

Uložení koordinátů pak vypadá následovně:

```
GRID*   6                               2.000000E+00
6.000000E+00
*           0.000000E+000
```

NASTRAN používá koncovky *.nas a *.bdf.

3.5.3 STEP

Souborový formát STEP (Standard for the Exchange of Product model data) je popsán normou ISO 10303-21.

Jedná se o jeden ze základních formátů využívaných v CAD technologiích, používá se pro práci s modely vytvořenými surface a solid modelováním.

Zde je ukázka souboru ve formátu STEP:

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(
/* description */ ('A minimal AP214 example with a single part'),
/* implementation_level */ '2;1');
FILE_NAME(
/* name */ 'demo',
/* time_stamp */ '2003-12-27T11:57:53',
/* author */ ('Lothar Klein'),
/* organization */ ('LKSoft'),
/* preprocessor_version */ ' ',
/* originating_system */ 'IDA-STEP',
/* authorization */ ' ');
FILE_SCHEMA (('AUTOMOTIVE_DESIGN { 1 0 10303 214 2 1 1}'));
ENDSEC;
DATA;
#10=ORGANIZATION('O0001','LKSoft','company');
#11=PRODUCT_DEFINITION_CONTEXT('part definition',#12,'manufacturing');
#12=APPLICATION_CONTEXT('mechanical design');
#13=APPLICATION_PROTOCOL_DEFINITION('','automotive_design',2003,#12);
#14=PRODUCT_DEFINITION('0',$,#15,#11);
#15=PRODUCT_DEFINITION_FORMATION('1',$,#16);
#16=PRODUCT('A0001','Test Part 1',&#39;&#39;,(#18));
#17=PRODUCT_RELATED_PRODUCT_CATEGORY('part',$,(#16));
#18=PRODUCT_CONTEXT(&#39;&#39;,#12,&#39;&#39;);
#19=APPLIED_ORGANIZATION_ASSIGNMENT(#10,#20,(#16));
#20=ORGANIZATION_ROLE('id owner');
ENDSEC;
END-ISO-10303-21;
```

STEP používá koncovky *.step a *.stp

Ze zápisu je vidět, že formát již není strukturou vůbec podobný kódu NASTRAN. Jak již bylo avizováno jedná se o úplně jiný systém úschovy dat, jde se totiž o CAD formát. Tj, model je vytvořen na základě solid nebo surface modelování, navíc STEP uchovává mnoho dalších informací nejen o modelu.

3.5.4 IGES

IGES (Initial Graphics Exchange Specification) začala vyvíjet skupina firem používající CAD technologie společně s ministerstvem obrany Spojených Státu již v roce 1978. Postupně se rozšířil do celého světa a je z něj jeden z nejpopulárnějších CAD formátu.

Zde je ukázka Souboru ve formátu IGES:

```
  S      1
1H,,1H;,4HSLOT,37H$1$DUA2:[IGESLIB.BDRAFT.B2I]SLOT.IGS;,                G      1
17HBravo3 BravoDRAFT,31HBravo3->IGES V3.002 (02-Oct-87),32,38,6,38,15,  G      2
4HSLOT,1.,1,4HINCH,8,0.08,13H871006.192927,1.E-06,6.,                    G      3
31HD. A. Harrod, Tel. 313/995-6333,24HAPPLICON - Ann Arbor, MI,4,0;      G      4
     116       1       0       1       0       0       0       0       1D      1
     116       1       5       1       0                               0D      2
     116       2       0       1       0       0       0       0       1D      3
     116       1       5       1       0                               0D      4
     100       3       0       1       0       0       0       0       1D      5
     100       1       2       1       0                               0D      6
     100       4       0       1       0       0       0       0       1D      7
     100       1       2       1       0                               0D      8
     110       5       0       1       0       0       0       0       1D      9
     110       1       3       1       0                               0D     10
     110       6       0       1       0       0       0       0       1D     11
     110       1       3       1       0                               0D     12
116,0.,0.,0.,0,0,0;                                                     1P      1
116,5.,0.,0.,0,0,0;                                                     3P      2
100,0.,0.,0.,0.,1.,0.,-1.,0,0;                                          5P      3
100,0.,5.,0.,5.,-1.,5.,1.,0,0;                                          7P      4
110,0.,-1.,0.,5.,-1.,0.,0,0;                                            9P      5
110,0.,1.,0.,5.,1.,0.,0,0;                                             11P      6
S      1G      4D     12P      6                                        T      1
```

IGES používá koncovky *.igs a *.iges

IGES je ve své podstatě podobný STEPu, ukládají se v něm geometrická data vytvořená nejčastěji v některé CAD aplikaci, využívající stejný způsob modelování objektů.

Kapitola 4

Vytvořené skripty

4.1 Import formátu NASTRAN

NASTRAN byl zvolen jako první právě proto, že jeho import nevyžaduje žádné rozšíření Blenderu či Pythnou. Formát reprezentuje data ve formě, která není pro Blener složitá na zpracování.

4.1.1 Skript

Celý zdrojový kód včetně komentářů se nachází na přiloženém CD, zde budou popsány pouze jeho důležité části a jejich činnost.

Skript tvoří tři hlavní části, první část složí ke zpracování dat ze souboru, druhá k vytvoření objektu ze získaných informací a poslední pak slouží k vytvoření položky pro import NASTRANu v menu Blenderu.

Následující výřez kódu definuje funkci pro vytvoření objektu v Blederu. Tato funkce je ve skriptu následně zavolána ve chvíli, kdy dojde ke zpracování potřebných dat.

```
def create_mesh(vertx, edgex, facex):

	sce = bpy.data.scenes.values()[0]
	mesh = bpy.data.meshes.new(filename)
	obj = bpy.data.objects.new(filename, mesh)


	verts = []
	for i in range(len(vertx)):
		verts.extend(vertx[i])

	edges = []
	for i in range(len(edgex)):
		edges.extend(edgex[i])

	faces = []
	for i in range(len(facex)):
		faces.extend(facex[i])

	smoothness = []
	for i in range(len(faces)):
		smoothness.append(False)

	# pridani geometrie
	mesh.add_geometry(int(len(verts)/3),int(len(edges)/2),int(len(faces)/4))
	# pridani uzlu do geometrie
	mesh.verts.foreach_set("co", verts)
	# pridani hran do geometrie
	mesh.edges.foreach_set("verts", edges)
	# pridani ploch do geometrie
	mesh.faces.foreach_set("verts_raw", faces)
	# pridani listu smoothness
	mesh.faces.foreach_set("smooth", smoothness)
	# nacteni sceny
	scene = bpy.data.scenes['Scene']
	#pridani meshe do objektu
	obj.data = mesh
	# pridani objektu do sceny
	scene.objects.link(obj)
```

Na začátku je vybrána scéna v Blenderu do které se bude importovat. V ní vytvoříme nový objekt pojmenovaný podle jména souboru. Tento objekt bude obsahovat jednu geometrii typu mesh, která ponese stejné jméno.

V druhé části se načtou získaná data do seznamů uzlů, hran, ploch a poslední seznam smoothness určuje systém vykreslení geometrie. Zda má hrana (přechod) mezi dvěma plochami vypadat ostře nebo má navozovat iluzi hladkého přechodu pomocí Phongova stínování.

V poslední části se do „meshe“ přidá geometrie pro uzly, hrany a plochy a do této

geometrie se doplní informace z již připravených seznamů.

Nakonec je ještě nutné načíst mesh do objektu a objekt přiřadit pracovní scéně.

```
for line in file:
    if line.startswith('GRID '):
        xyz.extend([float(line[24:32]),float(line[32:40]),float(line[40:48])])
        vert_id.append(int(line[8:16]))
    if line.startswith('GRID*'):
        count = 1
        xyz.extend([float(line[40:56]),float(line[56:72])])
        vert_id.append(int(line[8:24]))
    if (line.startswith ('*') and count == 1):
        count = 0
        xyz.append(float(line[8:24]))

    if line.startswith('CQUAD4'):
        quad_id.append(int(line[8:16]))
        geom_id.append(int(line[16:24]))
        for i in range(3,7):
            quad.append(int(line[i*8:(i+1)*8]))
    if line.startswith('CTRIA3'):
        tri_id.append(int(line[8:16]))
        geom_id.append(int(line[16:24]))
        for i in range(3,6):
            tri.append(int(line[i*8:(i+1)*8]))
    if line.startswith('CBAR'):
        bar_id.append(int(line[8:16]))
        geom_id.append(int(line[16:24]))
        for i in range(3,5):
            bar.append(int(line[i*8:(i+1)*8]))
file.close()
```

Ve výše uvedené sekci skriptu dochází k samotnému zpracování dat ze souboru. Nejprve se soubor otevře pro čtení, potom pomocí for cyklu postupujeme po jednotlivých řádcích. Podle toho jak začínají, řadí následující hodnoty do potřebných polí. Načítají se tedy souřadnice jednotlivých uzlů a dále informace o tom, ze kterých uzlů jsou složeny plochy, případně samostatné hrany.

Uložení všech hodnot v souboru je pevně určeno, proto je možné je načítat takto zvoleným způsobem. Jeden příklad za všechny, řádek s prvním slovem *GRID* je rozdělen následovně:

Znaky 1 až 8	jméno,
znaky 9 až 16	ID uzlu,
znaky 17 až 24	mezera,
znaky 25 až 32	souřadnice X,
znaky 33 až 40	souřadnice Y,
znaky 41 až 48	souřadnice Z.

V případě, že by formát neměl napevno daný system ukládání dat, bylo by nutné zajistit hledání informací jiným způsobem. Často využívaná možnost je detekce prázdných znaků (import některých formátů jako je například Wavefront). Tento způsob však nese svá úskalí. V ukládání souborů NASTRAN je možné, že na sebe informace navazují bez mezery(dlouhý zápis desetinného místa, atd.). Příklad zde:

```
GRID      423156        -735.999452.245751.00575
```

V takovém případě by se celá zadní část řádku načetla jako jedna souřadnice. Takový systém je proto u formátu NASTRAN nepoužitelný, alespoň ne v základní formě, s určitými úpravami by ho jistě bylo možné aplikovat.

Jak již bylo zmíněno dříve, vyskytuje se více možností uložení dat v souboru. Skript je na takový případ připraven a dokáže je správně zpracovat. Pokud tedy řádek začíná *GRID** uloží se do proměnné *count* hodnota 1 a u dalšího řádku se kontroluje, zda začíná znakem * a v proměnné *count* je uložena 1. Pokud tomu tak je, načte se z řádku souřadnice *Z* a proměnná *count* se opět vynuluje. K tomu dochází proto, že v další části souboru se mohou vyskytovat řádky uvozené stejným znakem, což by vedlo ke snaze načíst hodnotu která se v řádku nenachází.

Nyní se v paměti nachází několik polí. Ty nejdůležitější obsahují souřadnice uzlů, jejich ID, informace o plochách a také samostatných hranách.

Dále je nutno tato data zpracovat do podoby, ze které bude čerpat dříve definovaná funkce pro tvorbu objektu v Blenderu, tj. získat tři pole, která budou obsahovat korektně uložené informace o uzlech, o hranách mezi uzly a o plochách tvořících objekt.

```
for i in range(0,len(xyz),3):
        verts_list.append([xyz[i],xyz[i+1],xyz[i+2]])
```

Takto jednoduše uloží skript souřadnice jednotlivých uzlů do pole polí. Pole tedy bude vypadat asi takto: *[[x0,y0,z0],[x1,y1,z1],...]* Blenderu si z pole následně hodnoty načte a vytvoří podle nich v prostoru body odpovídající geometrii uložené v souboru.

Procedura potřebná k uložení pole obsahujícího body ploch je poněkud náročnější. Máme tedy dvě pole*quad* a *tri* obsahující ID uzlů, které tvoří plochy. Jak z názvu vyplývá, v jednom jsou uloženy trojúhelníky, v druhém pak čtyřúhelníky. Blender bere plochu jako pole 4 hodnot, pokud je poslední hodnota 0, jedná se o trojúhelník. Skript tedy musí z oněch dvou polí vytvořit jedno pole polí, které bude například ve tvaru *[[b1,b2,b3,b4],[b1,b2,b5,0],...]*. Problémů v takovém případě vyvstává několik.

Zaprvé, ID uzlů neodpovídá uložení souřadnic v poli *vert_list*, to skript řeší pomocí podobnosti polí. Tj. pokud je ID uzlu uloženo na čtvrté pozici v poli *vert_id*, pro plochu je použit její index, odpovídá totiž pozici v poli souřadnic.

Dalším problémem je, že první položka pole v Pythonu nese index 0. Tudíž pokud bude v modelu čtyřúhelník, jehož poslední bod bude uložen na první pozici v poli, bude si Blender myslet, že se jedná o trojúhelník. Tento nedostatek je vyřešen posunutím poslední hodnoty na začátek(pokud je rovna 0).

Pro zpracování pole trojúhelníků je postup oproti čtyřúhelníkům obohacen o přidání nuly na konec každého pole nesoucího informace o bodech tvořících trojúhelník.

Nyní zbývá sestavit pole jež bude určovat hrany v modelu. Musí tedy obsahovat jak samostatné hrany získané z dat v souboru, tak hrany, které se vytvoří mezi uzly při tvorbě ploch. Díky poli s plochami *face_l*, které již máme vytvořené dokážeme zjistit i tyto hodnoty. Trojúhelník má tři hrany, čtyřúhelník logicky čtyři. Získáme je propojením po sobě jdoucích hodnot v polích jednotlivých ploch, s tím, že poslední hrana vznikne spojením posledního a prvního bodu. Jediný nedostat je, že dvě sousedící plochy mají vždy společnou (alespoň jednu)hranu. Tudíž se v poli *help_edges* nachází některé hrany hned několikrát. Skript tedy pole projde po jednotlivých složkách a uloží je do nového pole *edges_l* s podmínkou, že pokud již danou hranu obsahuje, znovu ji nepřidá. Dostaneme tedy následující tvar: *[[b1,b2],[b2,b3],[b3,b1],....]*

Takto vznikly tři žádoucí pole, které již Blender dokáže jednoduše zpracovat.

Stačí tedy už jen zavolat funkci *create_mesh(verts_list, edges_l, faces_l)* a Blender vytvoří objekt, který byl vybrán k importu.

Poslední část skriptu registruje import formátu NASTRAN do menu Blenderu. Tato část je podobná pro všechny položky v menu, tudíž je převzata z importu formátu Wavefront(OBJ), pouze byla upravena do patřičné podoby.

```
class IMPORT_OT_nas(bpy.types.Operator):
    bl_idname = "import.nas"
    bl_label = "Import NAS"
    path = StringProperty(name="File Path", description="File path of NAS file", maxlen= 1024)

    def execute(self, context):
            load_nas(self.properties.path,context)

            return {'FINISHED'}

    def invoke(self, context, event):
            wm = context.manager
            wm.add_fileselect(self)
            return {'RUNNING_MODAL'}

menu_func = lambda self, context:self.layout.operator(IMPORT_OT_nas.bl_idname,text="Nastran(.nas)")

def register():
    bpy.types.register(IMPORT_OT_nas)
    bpy.types.INFO_MT_file_import.append(menu_func)

def unregister():
    bpy.types.unregister(IMPORT_OT_nas)
    bpy.types.INFO_MT_file_import.remove(menu_func)

if __name__ == "__main__":
    register()
```

Nejprve je deklarovaná třída *IMPORT_OT_nas*, která má dvě funkce, proveď (*execute*), která volá dříve naprogramovanou funkci k načítání souboru. Druhou funkcí je vyvolání okna s výběrem souborů(*invoke*).

Další dvě funkce *register* a *unregistr* dovolují registrovat, respektive odregistrovat položku v menu File - Import. Poslední dva řádky zaručují funkčnost v případě, že bude skript spuštěn ručně z text editoru.

4.2 Import formátu STEP a IGES

Jak již bylo zmíněno v předchozí kapitole, jedná se o formáty, které ukládají data pro Blender nepoužitelným způsobem. Proto je nutné data převést do tvaru, který software typu Blender bude schopný zpracovat. K tomu poslouží také již dříve zmiňovaná nadstavba Pythonu a to pythonOCC. S tím přichází avizovaný problém, a to nedostupnost knihovny pythonOCC pro Python 3.1, který je využit ve vyvíjené aplikaci. Skripty budou tedy naprogramovány pro starší verzi Blenderu 2.49, jež využívá vhodnou variantu

Pythonu.

4.2.1 Skripty

Importování forátů STEP a IGES je ve své podstatě shodné, proto jsou popsány ve společné kapitole. Import probíhá ve třech fázích. Nejprve je objekt načten ze souboru, následně je z něj vytvořen mesh, a v poslední fázi je z dat rekonstruován objekt v Blenderu.

Celé skripty jsou opět na přiloženém CD.

```
my_step_importer = STEPImporter(path)
my_step_importer.ReadFile()
shape = my_step_importer.GetCompound()
```

Díky této příkazové části dojde k načtení souboru ve formátu STEP. PythonOCC má importní modul pro několik formátů, STEP a IGES nevyjímaje. Následně skript do proměnné *shape* uloží tvar objektu.

```
my_iges_importer = IGESImporter(path)
my_iges_importer.ReadFile()
shape = my_iges_importer.GetCompound()
```

U formátu IGES je jediný rozdíl právě v této části. Ta se liší jen minimálně, a to použitím *IGESImporter* namísto *STEPImporter*, což je jeden z hlavních důvodu použití pythonOCC při hledání řešení importu těchto formátů.

Když dojde k načtení objektu, je nutné aplikovat algoritmus, který z něj vytvoří mesh strukturu. PythonOCC, je k tomuto účelu rovněž přizpůsoben a to hned několika funkcemi.

V případě těchto skriptů vypadá použití jedné z funkcí následovně:

```
BRepMesh().Mesh(shape,0.1)
```

První parametr určuje z čeho bude mesh vytvořen, tedy proměnná *shape*, do které byl dříve nahrán tvar importovaného objektu.

Druhý parametr funkce pak stanovuje přesnost, lépe řečeno odchylku s jakou bude kopírován tvar původního objektu. STEP či IGES objekt má díky svému stylu ukládání dat hladké plochy. V mesh struktuře to u rovných ploch není problém, ale u zaoblených tvarů je zapotřebí vytvořit hladký dojem vyšší hustotou meshe. To však může způsobit

nadměrné zatížení paměti, případně příliš hustý mesh, se kterým by se velice špatně pracovalo. Proto je zde možnost regulace. Pokud je parametr nastaven na hodnou 0.001, objekt se jeví velice hladce, oproti tomu hodnota 1 vytvoří mesh s řídkou topologií a nepřesným tvarem. Import ovšem proběhne o poznání rychleji. Proto je vhodné zvolit vhodný poměr přesnosti a času importu.

V poslední fázi je třeba vytvořit importovaný objekt v Blenderu. Nejprve dojde k rozdělení tvaru na menší části(entity) ze kterých byl vytvořen. První parametr opět určuje co se bude prohledávat, druhá pak, co se bude hledat. V tomto případě *FACE*, tedy plochy.

```
ex = TopExp_Explorer(shape, TopAbs_FACE)
```

Proměnná *ex* tak zahrnuje pole dat obsahujících informace o jednotlivých částech objektu. Dalším krokem je data zpracovat do patřičné podoby, tak aby je mohl Blender zpracovat. Tj, pole se souřadnicemi uzlů a pole s informacemi o bodech, které je tvoří. V tomto případě se jedná výhradně o trojúhelníky, jakožto nejmenší možný tvar plochy, který je navíc vždy rovný narozdíl od vícebodých polygonů.

Úkolem níže uvedeného cyklu je projít jednotlivé záznamy v poli *ex*, získat data o umístění uzlů a o plochách. Následně je uložit do polí *tab* respektive *tri*. Hodnoty z nich jsou zpracovány jako vstupní informace pro novou mesh strukturu v Blenderu.

```
while ex.More():
        mesh = Blender.NMesh.New(name)
        F = TopoDS().Face(ex.Current())
        L = TopLoc_Location()
        facing = (BRep_Tool().Triangulation(F,L)).GetObject()
        tab = facing.Nodes()
        tri = facing.Triangles()

        for i in range(tab.Lower(), tab.Upper()+1):
            x = tab.Value(i).Coord()
            pole.append(x)
            mesh.verts.append(Blender.NMesh.Vert(x[0],x[1],x[2]))

        for i in range(tri.Lower(), tri.Upper()+1):
            faceVertList = []
            y = tri.Value(i).Get()
            for k in range(0,3):
                faceVert = mesh.verts[y[k]-1]
                faceVertList.append(faceVert)
            mesh.addFace(Blender.NMesh.Face(faceVertList))
```

Nakonec je vytvořen objekt, k němu je přiřazen mesh a objekt je pomoci *link* spojen

se pracovní scénou. Cyklus se opakuje pro každou část objektu uloženo v proměnné *ex* dokud v ní nejsou žádné další záznamy.

```
ob = Blender.Object.New('Mesh', name)
ob.link(mesh)
scn = Blender.Scene.GetCurrent()
for o in scn.objects:
    o.sel = 0

scn.link(ob)
ob.sel = 1
ob.Layers = scn.Layers

ex.Next()
```

Jelikož se v každém opakování cyklu vytvoří nový objekt, bylo by dobré objekty spojit do jednoho celku, to zatím skript neumožňuje.

4.3 Test importu

U formátu NASTRAN proběhl test měření rychlosti importu. Několik souborů různé velikosti bylo postupně načítáno s následujícími výsledky:

1113 faces	0.59 sekundy
1548 faces	1.07 sekundy
3025 faces	2.83 sekundy
11119 faces	46.3 sekundy

Vliv na rychlost měl i systém uložení dat v souboru, velikostní rozdíl mezi první a druhou položkou není značný avšak časy se liší takřka o polovinu.

K porovnání s posledním importem proběhl test s objekt ve formátu OBJ, který obsahuje stejný počet ploch. Tento soubor trvá Blenderu importovat zhruba vteřinu, NASTRAN je však složitější formát a skript pro import Wavefront je v Blenderu obsažen již několik let.

Kapitola 5

Závěr

Cílem práce bylo vytvořit skripty pro import 3D dat do programu Blender na základě projektu MPO TIP 2009 „Inovace technologie výroby umělých kůží, ev.č. FR - TI1/266“.

Tým řešitelů vybral tři datové formáty, NASTRAN, IGES a STEP, jejichž import se povedlo realizovat. Import formátu NASTRAN je plně použitelný ve vyvíjené aplikaci. IGES a STEP zatím pouze pro Blender 2.49. I to má ale svůj význam, jelikož skripty pro verzi 2.5 mají z čeho vycházet. Pokud se podaří kompilace pythonOCC pro nejnovější verzi Pythonu, bude tvorba importních skritů IGES a STEP o hodně jednodušší. Nebude již třeba hledat způsoby, jak dostat CAD formát do mesh struktury, apod.

Z poslední části práce jasně vyplývá, že import datového typu NASTRAN by se dal zrychlit, k největšímu zdržení dochází při třídění dat do správné formy pro Blender. Samotné načtení dat u posledního testu totiž proběhlo za 0.34 sekundy. Pokud by skript data netřídil přes tolik pomocných polí, jistě by probíhal import rychleji. Z toho plyne jeden z postřehů pro další vývoj, tedy optimalizovat skript pro rychlejší nahrávání.

Také by jistě bylo dobré dodělat skripty importující IGES a STEP do požadované verze Blenderu. V tuto chvíli je přímý import těchto formátů pro vyvíjenou aplikaci nepoužitelný. V neposlední řadě by bylo vhodné zajistit rychlejší a kvalitnější výsledek „meshování“ CAD formátů, neboť tvorba mesh topologie pro složitější objekty vyžaduje velké množství paměti a trvá příliš dlouho.

Literatura

[1] cs.wikipedia.org/wiki/GNU_General_Public_License/

[2] cs.wikipedia.org/wiki/Python/

[3] cs.wikipedia.org/wiki/Blender/

[4] www.pythonocc.org/

[5] www.opencascade.org/

[6] docs.autodesk.com/

[7] http://www.blender.org/documentation/250PythonDoc/contents.html